Recovery of 3-D Motion and Structure by Temporal Fusion

Tarek M. Sobh and Kwangyoen Wohn

GRASP Laboratory

Department of Computer and Information Science

School of Engineering and Applied Science

University of Pennsylvania, Philadelphia, PA 19104

Tarek M. Sobh and Kwangyoen Wohn, "Recovery of 3-D Motion and Structure by Temporal Fusion", Proc. SPIE 1198, Sensor Fusion II: Human and Machine Strategies, 147 (March 1, 1990); doi:10.1117/12.969971;

Recovery of 3-D Motion and Structure by Temporal Fusion

Tarek M. Sobh and Kwangyoen Wohn

GRASP Laboratory
Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania, Philadelphia, PA 19104

ABSTRACT

We discuss the problem of recovering the 3-D motion and structure. An algorithm for computing the camera motion and the orientation of planar surface is developed. It solves for the 3-D motion and structure iteratively given two successive image frames. We further improve the solution by solving the ordinary differential equations which describe the evolution of motion and structure over time. The robustness of the entire process is demonstrated by the experiment with a moving camera which "flies" over a terrain model.

1 INTRODUCTION

The problem of recovering scene structure and the camera motion relative to the scene has been one of the key problems in computer vision. Many techniques have been developed for the estimation of structure and motion prameters (Tsai and Huang [2], Weng et al. [7] etc.). A lot of existing algorithms depend on evaluating the motion parameters between two successive frames in a sequence. However, recent research on structure and motion has been directed towards using a large number of frames to exploit the history of parametric evolution for a more accurate estimation and noise reduction (Ullman [3], Hildreth and Grzywacz [6], Iu and Wohn [8] etc.)

In this paper we describe a method for recovering the 3-D motion and orientation of a planar surface from an evolving image sequence. The algorithm utilizes the image flow velocities in order to recover the 3-D parameters. First, we develop an algorithm which iteratively improves the solution given two successive image frames. The solution space is divided into three subspaces - the translational motion, the rotational motion and the surface slope. The solution of each subspace is updated by using the current solution of the other two subspaces. The updating process continues until the motion parameters converge, or until no significant improvement is achieved.

Second, we further improve the solution progressively by using a large number of image frames and the ordinary differential equations which describe the evolution of motion and structure over time. Our algorithm uses a weighted average of the expected parameters and the calculated parameters using the 2-frame iterative algorithm as current solution and continues in the same way till the end of the frame sequence. Thus it keeps track of the past history of parameteric evolution. The system was tested on a sequence of images obtained by the motion of a camera over a planar surface.

2 MODELING

One can model an arbitary 3-D motion in terms of stationary-scene/moving-viewer as shown in Figure 1. The optical flow at the image plane can be related to the 3-D world as indicated by the following pair of equations originally derived by Longuet-Higgins and Prazdny [1], for each point (x, y) in the image plane :

$$v_x = \left\{x\frac{V_Z}{Z} - \frac{V_X}{Z}\right\} + \left[xy\Omega_X - \left(1+x^2\right)\Omega_Y + y\Omega_Z\right]$$

$$v_y = \left\{y\frac{V_Z}{Z} - \frac{V_Y}{Z}\right\} + \left[\left(1+y^2\right)\Omega_X - xy\Omega_Y - x\Omega_Z\right] ,$$

where v_x and v_y are the image velocity at image location (x, y), (V_X, V_Y, V_Z) and $(\Omega_X, \Omega_Y, \Omega_Z)$ are the translational and rotational velocity vectors of the observer, and Z is the unknown distance from the camera to the object.

For planar surfaces, the Z function is simply $pX + qY + Z_o$, where p and q are the planar surface orientations. The situation becomes, for each point, two equations in eight unknowns, namely, the scaled translational velocities V_X/Z_o, V_Y/Z_o and V_Z/Z_o, the rotational velocities Ω_X, Ω_Y and Ω_Z and the orientations p and q. Differential methods could be used to solve those equations by differentiating the flow field and by using approximate methods to find the flow field derivatives. The existing methods for computing the derivatives of the flow field usually do not produce accurate results. Our algorithm uses a discrete method instead, i.e, the vectors at a number of points in the plane is determined and the problem reduces to solving a system of nonlinear equations, a pair of equations represents the flow at each point as follows :

$$v_x = (1 - px - qy)\left(x\frac{V_z}{Z_o} - \frac{V_x}{Z_o}\right) + \left[xy\Omega_X - (1 + x^2)\,\Omega_Y + y\Omega_Z\right]$$

$$v_y = (1 - px - qy)\left(y\frac{V_z}{Z_o} - \frac{V_Y}{Z_o}\right) + \left[(1 + y^2)\,\Omega_X - xy\Omega_Y - x\Omega_Z\right]$$

It should be noticed that the resulting system of equations is nonlinear, however, it has some linear properties. The rotational part, for example, is totally linear, also, for any combination of two spaces among the rotational, translational and slope spaces, the system becomes linear. For the system of equations to be consistent, we need the flow estimates for at least four points, in which case there will be eight equations in eight unknowns.

3 TWO-FRAME ALGORITHM

The algorithm takes as input the estimate of the flow vectors at a number of points ≥ 4 obtained from motion between two images. It iterates updating the solution of each subspace by using the solution of the other two subspaces. Each update involves solving a linear system, thereby it requires to solve three linear systems to complete a single iteration. This process continues until the solution converges, or until no significant improvement is made. The algorithm proceeds as follows :

1. Set p, $q = 0$;
   input the initial estimate for rotation ;
   Solve the linear system for translation;
2. Use the translation and rotation from step 1 ;
   Solve the linear system for the slope ;
3. Set i=1;
   While (i $\leq$ Max. Iterations) and (no convergence) Do
   Solve for the rotations using latest estimates of translations, p and q;
   Solve for the translations using latest estimates of rotations, p and q;
   Solve for p, q using latest estimates of translations and rotations;
   end While ;

3.1 Complexity Analysis

As we mentioned earlier, one should notice in the equations relating the flow velocities with the slope, rotational and translational velocities that they are "quasi-linear" , if one can say so. The equations exhibit some linear properties. This suggests that a purely iterative technique for solving non-linear equations might not be an excellent choice, since, the variables are linearly related in some way. To think of a way of "inverting" the relations might be a good start, although to do that without a framework based on iterating and gravitating towards a solution is not a good idea.

This makes one think of applying a method which converges faster than a purely iterative scheme like Newton's method. However, the complexity of Newton's method is determined by the complexity of computing the inverse Jacobian, which is of an order of N^3, or $N^{2.81}$ multiplications as the lower bound using Strassen's technique. In our case, since we have at least 8 equations in 8 unknowns, the complexity is of order $8^3 = 512$ multiplications at every

iteration, and the method does not make any use of the fact that the set of equations at hand exhibits some linear properties.

The algorithm proposed, on the other hand, makes very good use of the fact that there are some linearity in the equations, by inverting the set of relations for each subspace at every iteration. The complexity at every iteration is of the order of the complexity of computing the psuedo-inverse which is of the order of ($3^3+3^3+2^3$) multiplications at each iteration, where the first 3 comes from solving the system for the rotational variables, the second 3 is for the translations, the last 2 is for p and q. This is equal to 62 multiplications at every iteration, which is significantly less than the 512 multiplications in a method like Newton's for example. It was noticed that the algorithm converged to solution in a very small number of iterations for most experiments we have conducted so far. The maximum number of iterations was 7.

Using the latest solution obtained from the two-frame analysis as the initial condition for the next two-frame problem in the image sequence would further decrease the complexity, as the next set of parameters would, most probably, be close in values to the current parameters, thus the number of iterations needed to converge to the new solution would decrease significantly.

3.2 Observations

- The algorithm is not sensitive to the initial condition of the orientation parameters. The plane is simply assumed to be a frontal one at the beginning. The slope parameters evolves with iterations.
- The algorithm is sensitive to input noise just like other existing algorithms, some experiments showes the sensitivity with respect to the change of viewing angle, table 3 includes some results of those experiments. Similarly, the algorithm performs better for a large number of points that are evenly distributed throughout the planar surface, than it does for clustered, smaller number of image points.
- It is proven that there exists dual solutions for such systems. However, if our method gravitates towards a "fixed point" in the solution space we can find the other explicitly in terms of the first one from the relations given by Waxman and Ullman [4].

4 MULTI-FRAME ALGORITHM

The ordinary differential equations that describe the evolution of motion and structure parameters are used to find the expression for the expected parameter change in terms of the previous parameter estimates. The expected change and the old estimates are then used to predict the current motion and structure parameters.

At time instant t, the planar surface equation is described by

$$Z = pX + qY + Z_o$$

To compute the change in the structure parameters during the time interval dt, we differentiate the above equation to get

$$\frac{dZ}{dt} = p\frac{dX}{dt} + X\frac{dp}{dt} + q\frac{dY}{dt} + Y\frac{dq}{dt} + \frac{dZ_o}{dt}$$

The time derivatives of (X, Y, Z) in the above expression are given by the three components of the vector $-(V+\Omega \times R)$ that represent the relative motion of the object with respect to the camera. Substituting these components for the derivatives and the expression $pX + qY + Z_o$ for Z we can get the exact differentials for the slopes and Z_o as

$$dZ_o = Z_o\left[(\Omega_Y + V_X)p - (\Omega_X - V_Y)q - V_Z\right]dt$$

$$dp = \left[p(\Omega_Y p - \Omega_X q) + (\Omega_Y + \Omega_Z q)\right]dt$$

$$dq = \left[q(\Omega_Y p - \Omega_X q) - (\Omega_X + \Omega_Z p)\right]dt$$

Using the above relations, we can compute the new structure parameters at time $t + dt$ as

$$\dot{p} = p + dp \ , \ \dot{q} = q + dq \ \text{ and } \ \dot{Z}_o = Z_o + dZ_o$$

Thus the slope parameters evolve at time $t + dt$ as follows :

$$\begin{bmatrix} \dot{p} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix} + \begin{bmatrix} \Omega_Y p - \Omega_X q & \Omega_Z & \Omega_Y \\ -\Omega_Z & \Omega_Y p - \Omega_X q & -\Omega_X \end{bmatrix} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} dt$$

The new translational velocity $\dot{V}$ at time $t + dt$ can be found in the absence of accelerations from

$$\dot{\mathbf{V}} = \mathbf{V} + \mathbf{V} \times \Omega dt$$

Dividing $\dot{V}$ by $\dot{Z}_o$ we get the new expected scaled translational velocity components at time $t + dt$ as follows :

$$\begin{bmatrix} \dot{V}_X \\ \dot{V}_Y \\ \dot{V}_Z \end{bmatrix} = \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} + \begin{bmatrix} -s & \Omega_Z & \Omega_Y \\ -\Omega_Z & -s & \Omega_X \\ \Omega_Y & -\Omega_X & -s \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} dt,$$

where s is expressed as follows :

$$s = (\Omega_Y + V_X)\, p - (\Omega_X - V_Y)\, q - V_Z$$

The expected rotational parameters at time $t + dt$ remain equal to their values at time t since

$$\dot{\Omega} = \Omega + \Omega \times \Omega dt = \Omega$$

and thus

$$\left(\dot{\Omega_X}, \dot{\Omega_Y}, \dot{\Omega_Z}\right) = (\Omega_X, \Omega_Y, \Omega_Z)$$

Our algorithm uses a weighted average of the expected parameters at time $t + dt$ from the above equations and the calculated parameters using the two-frame iterative algorithm as the solution at time $t + dt$, and continues in the same way until the end of the frame sequence. Thus it keeps track of the past history of parameteric evolution. The behaviour of the two-frame algorithm and the multi-frame algorithm can be conceptualized as a control system as shown in figures 2a and 2b.

5 RESULTS

The algorithm was run on a sequence of image data. The images were those of a planar surface being approached by a video camera mounted on a robot arm. The plane consisted of 120 dots. The sequence simulated the situation where an airplane approaches a runway for landing. One may think of the dots as lights to guide the airplane during a night landing. The actual rotational and translational velocities between each two subsequent shots were $\Omega_X = -3°$, $\Omega_Y = 0°$, $\Omega_Z = -5°$, $V_X = 0$ mm, $V_Y = 10$ mm and $V_Z = 20$ mm. To determine the flow vectors, the first order moments were used to calculate the center of mass of each one of the dots in the image sequence and then they were matched across the image sequence. Thus, there were 120 points at which the x and y displacements were available as the approximation to the flow velocities. In real image date, more elaborated flow recovery algorithm should be used in order to determine the flow field accurately.

Tables 1a and 1b are the recovered and actual parameters when the two-frame algorithm is used. Table 2 includes the parameters computed from the multi-frame algorithm, using the weighting factor = 1/2 . Table 3 includes the results of varying the view angle for the two-frame algorithm. It should be noted that some of the parameters improved significantly when we used the updating mechanism and kept track of the history of evolution of the motion and structure parameters. The translational velocities in the y and z directions and the plane orientation in the y direction were recovered more accurately at the end of the sequence using the second algorithm. However, error propagation caused a slight deterioration in the recovered values of a few parameters at the end of the experiment.

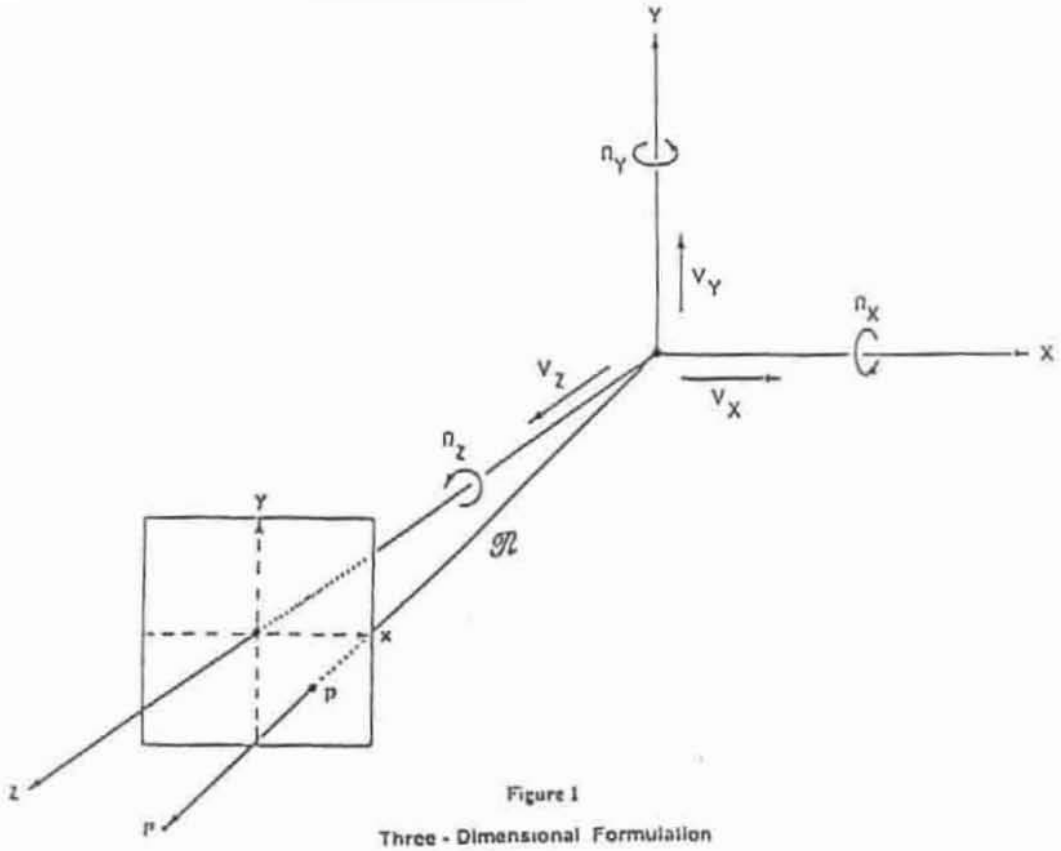


Figure 1

Three - Dimensional Formulation

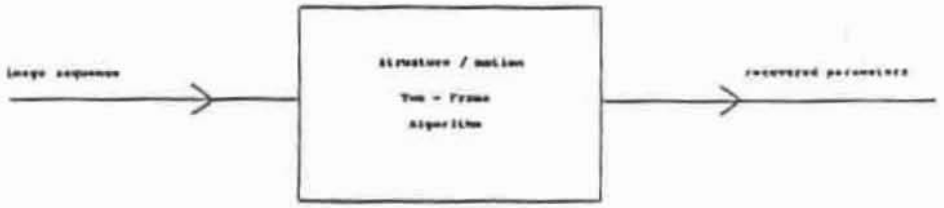


Figure 2a

Two - Frame Algorithm

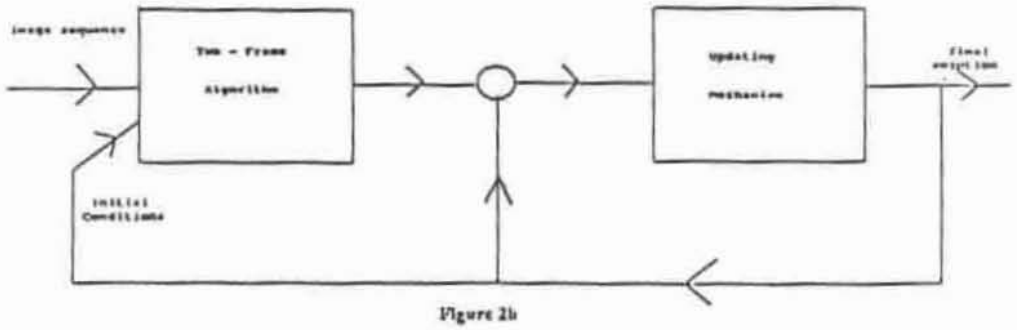
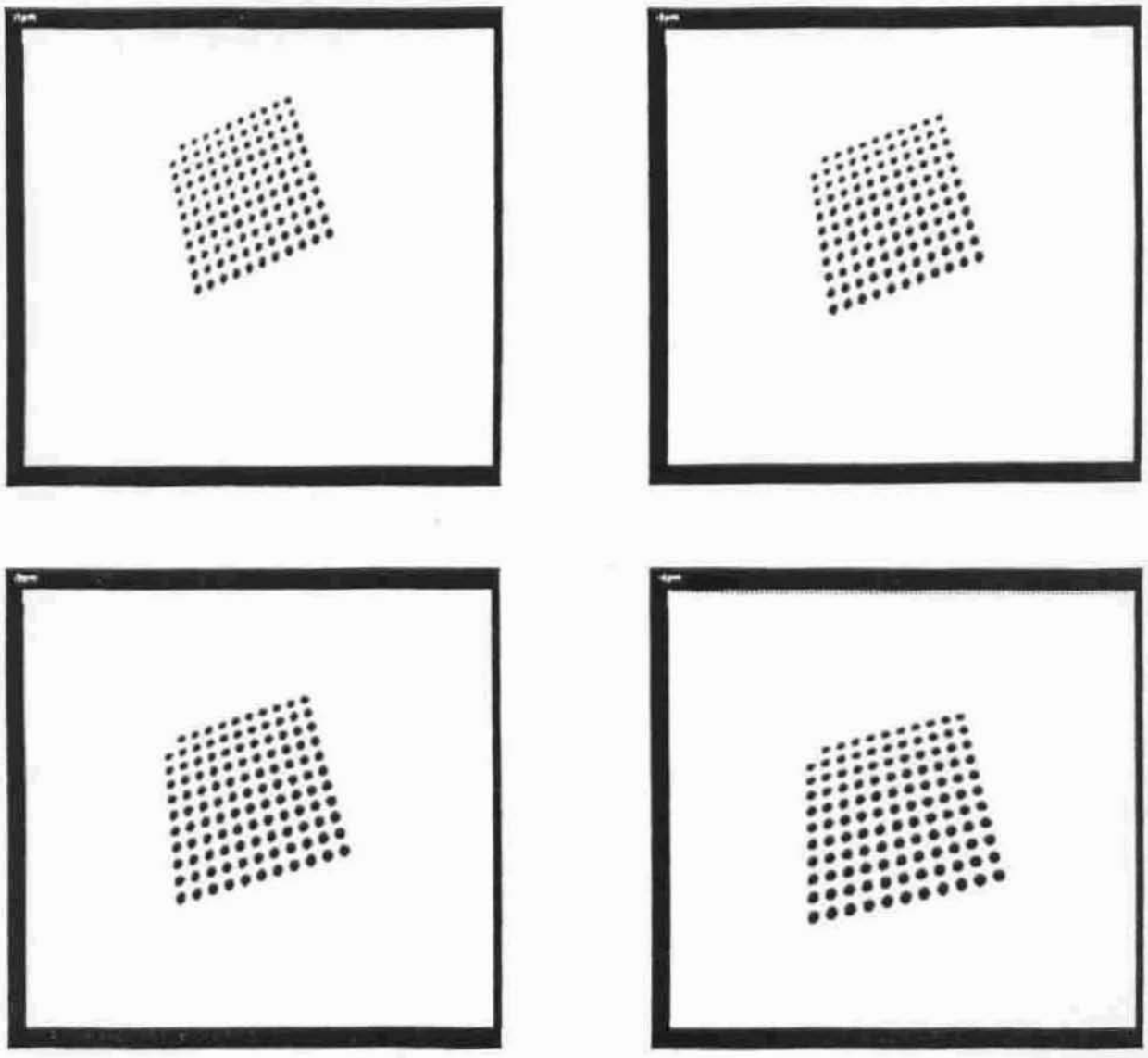


Figure 2b

Multi - Frame Algorithm

Updating Mechanism as a Control System

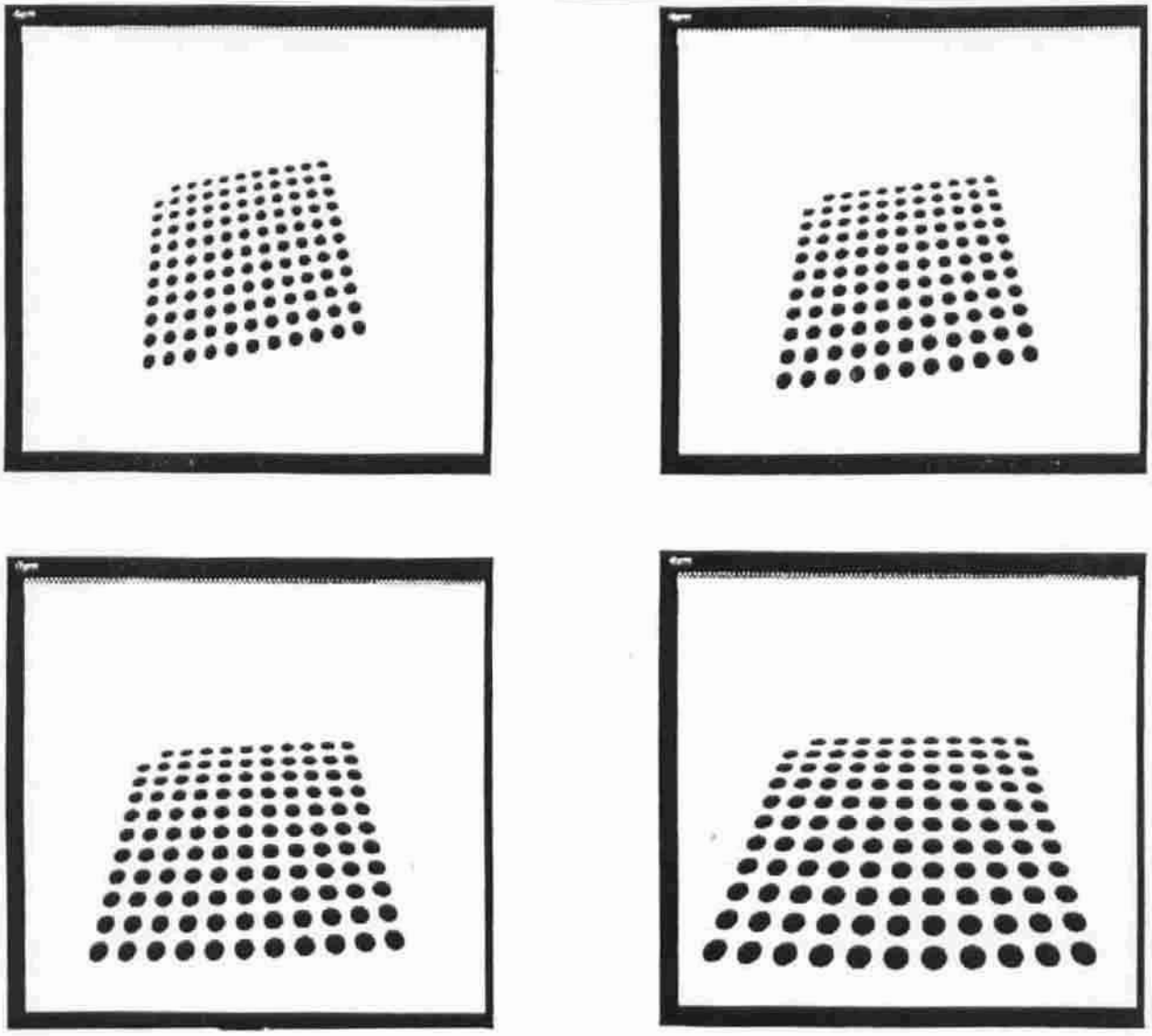
Experimental Results



The Image sequence (i1 -> i4)

Parameters	*I1 -> I2*	*Actual*	*I2 -> I3*	*Actual*	*I3 -> I4*	*Actual*	*I4 -> I5*	*Actual*
Ω_x	-0.096	-3.0	0.762	-3.0	0.429	-3.0	-0.224	-3.0
Ω_y	-0.162	0.0	-0.042	0.0	0.016	0.0	0.108	0.0
Ω_z	-2.537	-5.0	-2.889	-5.0	-2.82	-5.0	-3.005	-5.0
V_x	1.691	0.0	1.25	0.0	1.276	0.0	1.12	0.0
V_y	9.25	10.0	10.71	10.0	8.22	10.0	4.8	10.0
V_z	15.46	20.0	15.22	20.0	17.55	20.0	18.23	20.0
p	13.14	17.32	4.7	16.56	9.76	14.36	9.05	12.12
q	30.875	12.25	22.95	13.34	29.99	18.56	33.84	22.73

Table 1a
Recovered Parameters using the 2-Frame Algorithm



The Image sequence (i5 -> i8)

Paramelers	*I5 -> I6*	*Actual*	*I6 -> I7*	*Actual*	*I7 -> I8*	*Actual*
Ω_x	-0.136	-3.0	-2.9	-3.0	-3.04	-3.0
Ω_y	0.072	0.0	-3.7	0.0	0.159	0.0
Ω_z	-2.297	-5.0	-5.32	-5.0	-3.802	-5.0
V_x	0.806	0.0	2.16	0.0	0.7	0.0
V_y	2.963	10.0	5.54	10.0	7.167	10.0
V_z	18.53	20.0	11.35	20.0	16.934	20.0
p	17.94	13.67	22.04	10.23	15.07	4.42
q	13.57	37.38	25.04	45.51	71.06	78.36

Table 1b

Recovered Parameters using the 2-Frame Algorithm

Parameters	I1 -> I2	Actual	I2 -> I3	Actual	I3 -> I4	Actual	I4 -> I5	Actual
Ω_x	-0.1	-3.0	0.33	-3.0	0.384	-3.0	-0.08	-3.0
Ω_y	-0.214	0.0	-0.126	0.0	-0.05	0.0	0.0286	0.0
Ω_z	-2.64	-5.0	-2.67	-5.0	-2.8	-5.0	-2.99	-5.0
V_x	1.634	0.0	1.25	0.0	1.06	0.0	0.9	0.0
V_y	9.933	10.0	10.42	10.0	9.614	10.0	7.49	10.0
V_z	14.49	20.0	14.96	20.0	16.55	20.0	17.77	20.0
p	11.01	17.32	9.64	16.56	9.68	14.36	10.02	12.12
q	26.5	12.25	21.5	13.34	26.77	18.56	30.43	22.73

Parameters	I5 -> I6	Actual	I6 -> I7	Actual	I7 -> I8	Actual
Ω_x	-0.11	-3.0	-2.75	-3.0	-2.89	-3.0
Ω_y	0.045	0.0	-1.8	0.0	-0.8	0.0
Ω_z	-2.595	-5.0	-3.97	-5.0	-3.89	-5.0
V_x	0.697	0.0	1.124	0.0	0.78	0.0
V_y	5.47	10.0	6.85	10.0	8.31	10.0
V_z	18.68	20.0	15.5	20.0	17.76	20.0
p	14.86	13.67	25.34	10.23	16.1	4.42
q	22.34	37.38	28.04	45.51	74.86	78.36

Table 2

Recovered Parameters using the Many-Frames Algorithm

Parameters	Actual	38°	34°	29°	25°	20°	10°
Ω_x	-3.0	-3.01	-2.986	-2.787	-2.411	-1.923	2.483
Ω_y	0.0	0.16	0.149	0.165	0.176	0.318	0.015
Ω_z	-5.0	-3.78	-3.729	-3.521	-3.176	-1.983	-0.157
V_x	0.0	0.768	0.722	0.812	1.242	2.724	3.476
V_y	10.0	7.093	7.023	6.835	6.251	4.872	2.489
V_z	20.0	16.518	15.325	14.313	13.826	14.178	11.415
p	4.42	15.124	16.113	16.725	16.937	19.255	17.837
q	78.36	70.502	70.547	68.463	66.165	63.361	61.163

Table 3

Effect of Varying the View Angle on the Recovered Parameters (i7 -> i8)

6 DISCUSSION

The recovery method described here has a variety of applications. It can be useful in vision-guided applications such as autonomous landing and navigation. It may be a starting point for determining global structure - motion analysis of entire polyhedra, making it suitable for robotics applications in the "moving blocks world". Parallel implementations could be designed for such problems, thus solving for the structure - motion parameters for each surface separately. In fact solving the linear system at each iteration could also be parallelized. Extra processing will be needed to segment te image into separate planar surfaces.

We can further improve the solution by exploiting the temporal coherence of 3-D motion. We can develop the ordinary differential equations which describe the evolution of motion and structure in terms of the current motion/structure and the measurements (the 2-D motion vectors) in the image plane. As an initial step we can assume that the 3-D motion is piecewise uniform in time, The extended Kalman filter can then be used to update the solution of the differential equations.

7 REFERENCES

References

[1] H.C. Longuet-Higgins and K.Prazdny, *The interpretation of a moving Retinal Image*, Proc. Royal Society of London B, 208, 385-397.

[2] R.Y. Tsai and S.T. Huang, "Estimating three-dimensional motion parameters of a rigid planar patch", *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-29(6), December 1981.

[3] S. Ullman, *Maximizing Rigidity: The incremental recovery of 3-D structure from rigid and rubbery motion*, AI Memo 721, MIT AI lab. 1983.

[4] A.M. Waxman and S. Ullman, *Surface Structure and 3-D Motion From Image Flow: A Kinematic Analysis*, CAR-TR-24, Center for Automation Research, University of Maryland, October 1983.

[5] M. Subbarao and A.M. Waxman, *On The Uniqueness of Image Flow Solutions for Planar Surfaces in Motion*, CAR-TR-113, Center for Automation Research, University of Maryland, April 1985.

[6] N.M. Grzywacz and E.C. Hildreth, *The Incremental Rigidity Scheme for Recovering Structure from Motion: Position vs. Velocity Based Formulations*, MIT A.I. Memo No. 845, October 1985.

[7] J. Weng, T.S. Huang and N. Ahuja, "3-D Motion Estimation, Understanding and Prediction from Noisy Image Sequences", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(3), May 1987.

[8] S-L. Iu and K. Wohn, "Estimation of 3-D Motion and Structure Based on a Temporally Oriented Approach with the Method of Regression", *IEEE Workshop on Visual Motion*, March 1989, Irvine, CA, 273-281.